SELECTING THE t^{th} LARGEST USING BINARY ERRORLESS COMPARISONS*

by

Abdollah Hadian and Milton Sobel

Technical Report No. 121

University of Minnesota

May 1969

* This research was supported by National Science Foundation Grant 9018.

1. Introduction.

There are given n numbers, or items with scalar attributes, $x_1, x_2, \ldots, x_n$ which are unknown and assumed to be pairwise unequal. For given t, we wish to find the t^{th} largest of these numbers using only binary errorless comparisons; each comparison between two x's tells us only which is larger and which is smaller.

Two criteria are considered for evaluating and comparing procedures that accomplish our goal. A minimax (or M-optimal) procedure minimizes the maximum number of comparisons needed to find the t^{th} largest. Assuming a random ordering at the outset (with equal probability for each of the $n!$ arrangements), an E-optimal procedure is one that minimizes the expected number of comparisons required. Since the same procedure in general does not satisfy both criteria, we may refer to these two criteria as separate problems, the M-problem and the E-problem.

Kislicyn [2] considers the M-problem and gives an upper bound $P_t(n)$ for the number of comparisons $M_t(n)$ required by an M-optimal procedure, namely

$$(1.1) \qquad M_t(n) \leq P_t(n) = n - 1 + \sum_{i=1}^{t-1} [\log(n-i)],$$

where $[x]$ is the usual notation for the integer part of x and all logarithms are to the base 2. It is well known that $M_1(n) = n - 1$ and many procedures, including the knock-out tournament, achieve this lower bound. Kislicyn's work is also discussed by Moon [3 - page 48].

Two problems related to ours are the problem of ordering the t largest and the problem of selecting the t largest (unordered)

considered by Sobel in [6] and [7], respectively. For ordering the 2 largest items the best possible M-value $M_2^{(0)}(n)$ was already found by Schreier [4] and Slupecki [5], namely

$$(1.2) \qquad M_2^{(0)}(n) = n - 1 + [\log(n-1)] = P_2(n).$$

Since we cannot find the 2^{nd} largest item without also finding the largest item, it follows that $M_2(n)$ is also given by (1.2). Sobel gives several procedures for ordering the 2 largest items in [6], two of which are M-optimal, and it follows that all these procedures can also be used in our problem for $t = 2$; the ordering of the t best is also considered in [1]. The problem of selecting the 2 best (unordered) in [7] is related to our problem for both $t = 2$ and $t = 3$ since in either case we know which are the two largest items at termination. Hence the Min $\{M_2(n), M_3(n)\}$ is an upper bound for the M-value for the $t = 2$ selection problem. Conversely the best possible M-value given by

$$(1.3) \qquad M_2^{(S)}(n) = n - 1 + [\log(n-2)]$$

for the $t = 2$ selection problem in [7] is a lower bound for both $M_2(n)$ and $M_3(n)$ in our problem. The relationship of our problem to other research problems is considered in Section 6.

In this paper we improve on $P_t(n)$ in (1.1) by introducing three new procedures R_1, R_2 and R_3 which yield new sharper upper bounds $M_t(n|R_i)$ $(i = 1, 2, 3)$ of $M_t(n)$ such that for all t and n

$$(1.4) \qquad M_t(n) \le M_t(n|R_3) \le M_t(n|R_2) \le \text{Min}\left(P_t(n), P_{n-t+1}(n)\right)$$

and for $t < [\frac{n+1}{2}]$ we have $M_t(n|R_1) \le P_t(n)$.

One example for $N = 8$, $t = 4$ (which also affects the result for $n = 9$, $t = 5$) requires at most 12 comparisons $< M_4(8|R_3) = 13$ and shows that even our new bounds are not the best possible. However this counterexample is an isolated phenomenon and we have found that $M_t(n|R_3)$ is difficult to improve upon for most values of n and t. In fact, for an infinite sequence of n values and $t = (n + 1)/2$, procedures R_2 and R_3 are identical and are conjectured to be both M-optimal and E-optimal.

In addition, expressions are derived for the expected number of comparisons $A_t(n|R_1)$ under procedure R_1.

2. Procedure R_1.

The basic ideas of the procedure are the use of a binary expansion, the use of recursion methods, and the simple fact that the largest among any $n - t + 2$ items has at most $t - 2$ items above it and hence can be eliminated as a competitor for the position of t^{th} largest in the entire set of n (which has exactly $t - 1$ items above it).

Let the binary decomposition of $n - t + 2$ be given by

$$(2.1) \qquad n - t + 2 = 2^{r_1} + 2^{r_2} + \ldots + 2^{r_s},$$

where the integers $r_1 > r_2 > \ldots > r_s \geq 0$ and $s \geq 1$ are defined by (2.1). We note that (2.1) can also be written as

$$(2.2) \qquad r_i = [\log(n - t + 2 - \sum_{j=1}^{i-1} 2^{r_j})] \qquad (i = 1, 2, \ldots, s).$$

The procedure R_1 is described in four steps as follows.

1. Choose $n - t + 2$ items randomly and divide them into s groups, the group size for the i^{th} group being 2^{r_i} as indicated by

(2.1). Find the largest in each group, using the well-known knock-out tournament method. (The remaining $j - 2$ items are temporarily held in reserve.)

2. Find the largest among these $n - t + 2$ by comparisons among these s winners. In doing this we start with the smallest group (of size 2^{r_s}) and compare it's winner with that of the second smallest, then compare the largest of both groups with the largest in the third smallest group, etc. This gives a connected graph (or a tree) with $n - t + 2$ vertices containing a unique maximal element, which we denote by $x^{(i)}$ if it comes from the i^{th} group.

3. Since $x^{(i)}$ cannot be the t^{th} largest of the entire set, we remove it from our graph along with all the line segments connecting it to other vertices; this gives us a number of unconnected subsets. Using exactly one of the $t - 2$ items in reserve, we rebuild another monolithic connected structure (or tree) of size $n - t + 2$. The largest is again removed and this step is repeated until all $t - 2$ reserve units are used up.

4. We then conduct a simple play-the-winner tournament (eliminating all losers) among all competitors for 2^{nd} place in the final structure with $n - t + 2$ items. The winner is the t^{th} largest in the original set of n items.

Derivation of $M_t(n|R_1)$.

The maximum number of line segments (or connections) are lost in step 3 when $i = 1$, i.e., the worst case is when $x^{(i)} = x^{(1)}$ comes from the first group of size r_1. This results in a loss of $r_1 + 1$ connections if $s > 1$ or r_1 if $s = 1$ (which we write as $r_1 + 1 - \delta_{1s}$ for general s). Hence we need at most $r_1 + 1-\delta_{1s}$

comparisons to rebuild another tree. Using the notation $M(t, n)$ for the number of comparisons required in step 3 above, we obtain the recursion

$$(2.3) \qquad M(t, n) = r_1 + 1 - \delta_{1s} + M(t - 1, n - 1) \quad \text{for} \quad t - 2 > 0$$

with the boundary condition given by $M(2, n) = 0$. We note that for $M(t - 1, n - 1)$ on the right side of (2.3) the value of $n - 1 - (t - 1) + 2$ is the same as for $M(t, n)$ and hence the r's defined in (2.1) do not change. From (2.3) by iteration we easily obtain

$$(2.4) \qquad M(t, n) = (t - 2)(r_1 + 1 - \delta_{1s}).$$

To obtain $M_t(n|R_1)$ from (2.4) we have to add on exactly $n - t + 1$ comparisons for steps 1 and 2 and at most $r_1 - \delta_{1s}$ for step 4. This gives

$$(2.5) \qquad M_t(n|R_1) = n - t + 1 + (t - 2)(r_1 + 1 - \delta_{1s}) + r_1 - \delta_{1s}$$

$$= n - 1 + (t - 1)(r_1 - \delta_{1s}).$$

Using (2.2) with $i = 1$, it is readily observed that for all integers $s \geq 1$

$$(2.6) \qquad r_1 - \delta_{1s} = [\log(n - t + 1)] = \{\log(n - t + 2)\} - 1$$

where $\{x\}$ is the smallest integer not less than x. Hence we obtain from (2.5)

$$(2.7) \qquad M_t(n|R_1) = n-1+(t-1)[\log(n-t+1)] = n-t+(t-1)\{\log(n-t+2)\}.$$

Remark. It should be noted that the total number of comparisons $M_t(n|R_1)$ also satisfies the recursion (2.3) if we take $t \geq 1$ and change the

boundary condition so that $M_1(n|R_1) = n-1$. We make use of this observation in writing the recursion formulas (4.1) for a new procedure R_3 in section 4.

The upper bound for $M_t(n)$ found by Kislicyn [2] is

$$P_t(n) = n - 1 + \sum_{i=1}^{t-1} [\log(n-i)]. \tag{2.8}$$

Comparing this with the middle expression in (2.7), we note that our expression is obtained from (2.8) by replacing each term in the summation by the smallest summand. Hence

$$P_t(n) - M_t(n|R_1) = \sum_{i=1}^{t-2} ([\log(n-i)] - [\log(n-t+1)]) \geq 0. \tag{2.9}$$

For certain subsequences of n-values the difference in (2.9) approaches zero and for others (as in (2.12) below) it approaches infinity.

We now consider a special sequence of n values and t values

$$n = 2^{r+1} - 3, \qquad t = 2^r - 1 \tag{2.10}$$

which is indexed by r ($r = 1, 2, \ldots$). By (2.7) we find that

$$M_t(n|R_1) = (r + 1)(2^r - 2) \tag{2.11}$$

and the difference in (2.9) is

$$P_t(n) - M_t(n|R_1) = 2^r - 3 \to \infty \tag{2.12}$$

as $r \to \infty$. It is interesting to note that for each r-value in (2.10), the procedure R_1 has no variation in the number of comparisons required, i.e., the minimum number of comparisons is also given by (2.11) (see also remark after (2.15)).

Derivation of $A_t(n|R_1)$.

To derive the expected number of comparisons under procedure R_1 we again consider steps 1 and 2, step 3, and step 4 separately. For steps 1 and 2 we again use $n - t + 1$ comparisons. In step 3 we use the fact that the probability that the i^{th} group yields the largest of $n - t + 2$ is $p_i = 2^{r_i}/(n-t+2)$ $(i = 1, 2, \ldots, s)$. The number of line connections broken by removing $x^{(i)}$ is $r_i + i - \delta_{is}$, since we have r_i connections to $x^{(i)}$ within the i^{th} group, we pick up one connection for each of the $i - 1$ larger groups, and we pick up $1 - \delta_{is}$ connections from the smaller groups. In step 4 we have to find the best one of $r_i + i - \delta_{is}$ competitors and hence we use $r_i + i - 1 - \delta_{is}$ comparisons. At each iteration the items are associated with the group they came from and the new item is associated with the depleted i^{th} group. The marginal probability that the i^{th} group yields the largest remains the same at each iteration and we obtain

$$(2.13) \qquad A_t(n|R_1) = n-t+1+(t-2)\sum_{i=1}^{s} p_i(r_i+i-\delta_{is}) + \sum_{i=1}^{s} p_i(r_i+i-1-\delta_{is})$$

$$= n-t+(t-1)\sum_{i=1}^{s} p_i(i+r_i-\delta_{is})$$

Since

$$(2.14) \qquad r_i - \delta_{is} = [\log(n-t+1 - \sum_{j=1}^{i-1} 2^{r_j})] = \{\log(n-t+2 - \sum_{j=1}^{i-1} 2^{r_j})\} - 1,$$

it follows from (2.11) that

$$(2.15) \qquad A_t(n|R_1) = n - 1 + (\frac{t-1}{n-t+2}) \sum_{i=1}^{s} 2^{r_i}(i + \{\log \sum_{j=i}^{s} 2^{r_j}\}).$$

For the subsequence (2.10) we find that $A_t(n|R_1)$ reduces to (2.11); this also follows from the above observation that there is no variation when procedure R_1 is used for any r-value in (2.10).

3. Symmetrized Version of Procedure R_1.

It should be noted that procedure R_1 is asymmetric with respect to the median and is generally better for $t \leq [\frac{n+1}{2}]$ than for $t > [\frac{n+1}{2}]$; for example, $M_2(4|R_1) = 4 < M_3(4|R_1) = 5$. In exceptional cases the reverse holds; for example $M_3(6|R_1) = 9 > M_4(6|R_1) = 8$. In this section we define a symmetrized version R_2 of procedure R_1 which takes advantage of these inequalities.

A procedure R_1' dual to R_1, is first defined by interchanging t and $n - t + 1$ everywhere. Corresponding to equation (2.1) we write

$$(3.1) \qquad t + 1 = 2^{r_1'} + 2^{r_2'} + \ldots + 2^{r_s'},$$

and put $n - t - 1$ items in reserve. We find the smallest in each group and, starting with the smallest group, find the smallest of all $t + 1$ items. Then we eliminate items with t or more superiors.

We define the symmetrized procedure R_2 by using either R_1 or R_1', whichever gives a smaller maximum, This clearly gives the result

$$(3.2) \quad M_t(n|R_2) = \text{Min}\left(M_t(n|R_1), M_t(n|R_1')\right)$$
$$= \text{Min}\left(M_t(n|R_1), M_{n-t+1}(n|R_1)\right).$$
$$= \text{Min}\left(n-1+(t-1)[\log(n-t+1)], n - 1 + (n-t)[\log t]\right)$$

We wish to show that for r large the rows of table 1 for n close to but not greater than 2^{r+1} form an arithmetic progression (AP) with common difference r for all values of t except at most a few close to n/2. Under procedure R_1 for any fixed $n = 2^r + x$ with $0 < x \leq 2^r$, it is easy to show (proof omitted) that for t running from 1 to x the values of $M_t(n|R_1)$ form an AP with common difference r. In

particular, for $n = 2^{r+1}$ the AP extends up to $t = 2^r$.

Under procedures R_2 this property does not hold exactly but we wish to show that the same property holds asymptotically as $r \to \infty$. Let $n = \theta 2^r$ for fixed θ with $1 < \theta \leq 2$ and let $t = \epsilon 2^r$ for fixed ϵ with $0 < \epsilon \leq 1$. In order that $M_t(n|R_1)$ be not greater than $M_t(n|R_1')$, we can use the last expressions in (3.2) to obtain for asymptotically large r

$$(3.3) \qquad (\epsilon 2^r - 1)[\log 2^r(\theta - \epsilon)] \leq 2^r(\theta - \epsilon)[\log(\epsilon 2^r)]$$

We note from (3.3) that the inequality holds for $r \to \infty$ both for $\epsilon < \theta/2$ and for $\epsilon = \theta/2$. In particular, for $\theta = 2$ it holds for $1 \leq t \leq 2^r$. Hence for $r \to \infty$ the values of $M_t(n|R_2)$ are asymptotically the same as $M_t(n|R_1)$ for all $t < \theta 2^{r-1} = n/2$. In particular, for $\theta = 2$ the length of the AP in the row for $n = 2^{r+1}$ divided by the length 2^r of the whole row tends to 1 as $r \to \infty$.

We note from Table 1 that procedure R_3 defined in the next section affects only a small portion of the entries and, based on numerical studies, it is conjectured that the same properties proved above also hold for procedure R_3.

4. An Improved Procedure, R_3.

The basic idea is to give us the flexibility whenever possible of deciding at each stage of the algorithm whether we wish to eliminate the next item off the top (because it has $n - t + 1$ or more inferiors) or off the bottom (because it has t or more superiors). For $n - t + 2 \leq 4$ the break-up in step 3 of procedure R_1 gives us structures with at most 2 items connected; hence we are free to eliminate the next

item from the bottom. Conversely, for $n - t + 2 \geq 7$ the break-up may contain structures of size 3, 4 or larger which are oriented to finding the largest and we then have to continue eliminating off the top. In between, for $n - t + 2 = 5$ or 6, we have a similar flexiblity if and only if the removal of $x^{(i)}$ breaks up the structure of size 4; our formulas below reflect his complication for $n - t + 2$ equal to 5 or 6. A similar discussion holds for changes of strategy in the reverse direction if we replace $n - t + 2$ above by $t + 1$.

Let $f_{t,n}$ (and $f'_{t,n}$) denote the maximum number of comparisons under the new procedure R_3 if we eliminate the first item off the top (off the bottom). Then $M_t(n|R_3) = \min(f_{t,n}, f'_{t,n})$. The recursion formulas that define procedure R_3 can now be written as six equations but, since the second three equations are dual to the first three, we write only the latter as follows:

$$
\begin{aligned}
f_{t,n} &= 2 + M_{t-1}(n-1|R_3) && \text{for } n-t = 1 \text{ or } 2 \\
f_{t,n} &= \max(3 + M_{t-1}(n-1|R_3),\ n-t-2+f_{t-1,n-1}) && \text{for } n-t = 3 \text{ or } 4 \qquad (4.1) \\
f_{t,n} &= \{\log(n-t+2)\} + f_{t-1,n-1} \ . && \text{for } n-t \geq 5
\end{aligned}
$$

The three dual equations are obtained by interchanging $f_{t,n}$ and $f'_{t,n}$ and replacing any t which is not a subscript by $n - t + 1$. Since $f_{t,n}$ now represents the total number of comparisons, rather than the middle part as in (2.3), the boundary conditions are

$$f_{1,n} = n - 1 = f'_{1,n}; \quad f_{t,t} = t - 1 \qquad (4.2)$$

for all n and t.

The results of this iteration are given in Table 1 for $n \leq 20$. In Table 1 the cases for which procedure R_3 gives an improvement over procedure R_2 are shown by a star on the appropriate entry.

It is interesting that for the special subsequence defined in (2.10) we have $n - t + 2 = t + 1 = 2^r$ and the equations (4.1) reduce to (2.3) so that

$$(4.3) \qquad M_t(n|R_1) = M_t(n|R_2) = M_t(n|R_3) = (r+1)(2^r - 2)$$

and since there is 'no variation' in the results for these cases the common value in (4.3) is also the expectation for all 3 procedures.

5. A Counterexample.

To show that the results of procedure R_3 can be further improved in isolated cases we now give a procedure due to Doren for the special case $n = 8$ and $t = 4$; the only other case that this appears to affect is that of $n = 9$ and $t = 5$. The idea behind this procedure is to find a strategy that allows us to simultaneously eliminate items 'off the top' and 'off the bottom'; this appears to be possible only at one stage and only for $n = 8$ and $t = 4$.

We separate the 8 items into 2 groups of 4 and find the best item (by knock-out) in each group. Let the 2 groups be denoted by

$$(5.1) \qquad \begin{aligned} &x_1 < x_2 < x_4, \quad x_3 < x_4; \\ &y_1 < y_2 < y_4, \quad y_3 < y_4 \end{aligned}$$

We compare x_2 and y_2 and suppose (without loss of generality) that $x_2 < y_2$. This simultaneously eliminates x_1 and y_4 from contention and we need the 3rd largest of the remaining six. Compare x_2 and y_3; in the worst case $x_2 < y_3$ and we eliminate x_2 so that we now need

the 3rd largest of 5 items with the relations $y_1 < y_2$ and $x_3 < x_4$. Since $M_3(5) = 6$ under any of our procedures and we use both relations, we need exactly 4 more comparisons. Thus we have a total of 6 (from (5.1)) + 2 + 4 = 12 comparisons. All of our procedures R_1, R_1', R_2 and R_3 require 13 comparisons for this case.

Using the recursion (2.3), this result also gives a reduction of one for the case $n = 9$, $t = 5$, but does not affect any more cases.

6. Some Related Problems and Associated Future Research.

To show that our problem of finding the t^{th} largest is central to several other related problems, we briefly mention some of these.

I. Selecting the t largest of n items.

Here we consider the procedure of putting 1 unit aside (call it x), find the t^{th} largest (say y) of the remaining $n - 1$, and finally compare x and y (let z denote the larger). Then the union of z and the $t - 1$ units found to be larger than y constitute the t largest (unordered). If we use the basic procedure R_1 with $M_t(n)$ given by (2.7) then the maximum value $M_t^{(S)}(n)$ for the selection problem is

$$(6.1) \qquad M_t^{(S)}(n) = n - 1 + (t-1)[\log(n-t)].$$

It is conjectured that if we use an M-optimal procedure for finding the t^{th} largest of $n - 1$ then the resulting procedure (using the above) will be an M-optimal selection procedure.

II. For a fixed number of comparisons C ($C > t$) find a subset of the n items such that (i) it contains the t largest (t given), and (ii) its maximum (or expected) size is as small as possible.

Using the expression for $M_t(n|R_1)$ in (2.7), we find the smallest integer x such that

$$(6.2) \qquad x - 1 + (t-1)[\log(x-t+1)] \leq C;$$

for $C > t$ the solution $x \geq t + 1$ is unique since the left side of (6.2) is (strictly) increasing in x. Then we can find the t largest in a randomly chosen subset of size x. Combining these t with the remaining $n - x$ units gives a subset which is proposed as a solution for II.

It is conjectured that this procedure will have some optimal properties, at least when equality holds in (6.2). For example, if $n = 2^{r+1} - 3$, $t = 2^r - 1$ and $C = (r+1)(2^r - 2)$, then the above procedure gives a subset of size exactly t, i.e., the minimum and maximum subset size are both t.

III. Find and order the t largest of a set of n items, for t, n both given and $1 \leq t \leq n/2$.

Among the possible procedures we can find the t^{th} largest by starting with one of our procedures (say R_1) and then completely ordering the $t - 1$ items which we know to be above it; the latter can be accomplished by one of the several procedures considered in [1] and [6]. This procedure gives good results for small values of t; in particular, it is optimal for $t = 2$ and appears to be optimal for $t = 3$, although the result for $t = 3$ is not proved.

Acknowledgement.

The authors wish to thank David G. Doren of Intech Inc., Minneapolis, Minnesota for contributing the example given in Section 5 and for some stimulating conversations on this research.

Table 1

The Maximum Number of Comparisons Under Procedure R_3

n \ t	1	2	3	4	5	6	7	8	9	10
2	1									
3	2	3								
4	3	4								
5	4	6	6§							
6	5	7	8							
7	6	8	10	11*						
8	7	9	11	13+						
9	8	11	12	14	16+					
10	9	12	14*	15	17					
11	10	13	16	18*	18	20				
12	11	14	17	20	23	21				
13	12	15	18	21	24	26	24§			
14	13	16	19	22	25	28	27			
15	14	17	20	23	26	29	30	35		
16	15	18	21	24	27	30	33	36		
17	16	20	22	25	28	31	34	37	40	
18	17	21	24*	26	29	32	35	38	41	
19	18	22	25*	30	30	33	36	39	42	45
20	19	23	27	30*	35	34	37	40	43	46

* These six entries are the ones for which procedure R_3 improve upon the result of procedure R_2; in each case there was a saving of exactly one.

\+ These two entries can be reduced by one if we use the counterexample explained in the text in section 5.

§ These two entries correspond to $r = 2$ and 3 in (2.10); in each of these cases the maximum and minimum number of comparisons are equal under any of our procedures.

References

[1] Hadian, A. (1969). Optimality properties of various procedures for ranking n different numbers using only binary comparisons. Technical Report No. 117, Department of Statistics, University of Minnesota.

[2] Kislicyn, S. S. (1964). On the selection of the k^{th} element of an ordered set by pairwise comparison (Russian). Sibirsk. Mat. Ž. 5 557-564 (MR 29 No. 2198).

[3] Moon, J. W. (1968). Topics on Tournaments, Holt, Rinehart and Winston, New York.

[4] Schreier, J. (1932). On tournament elimination systems (Polish). Mathesis Polska 7 pp. 154-160.

[5] Slupecki, J. (1949-51). On the system S of tournaments. Colloq. Math. II pp. 286-290.

[6] Sobel, M. (1968). On an optimal search for the t best using binary errorless comparisons: The ordering problem. Technical Report No. 113, Department of Statistics, University of Minnesota.

[7] Sobel, M. (1968). On an optimal search for the t best using binary errorless comparisons: The selection problem. Technical Report No. 114, Department of Statistics, University of Minnesota.